REMARKS

In view of the foregoing amendments and following remarks responsive to the Office Action of October 21, 2005, Applicant respectfully requests favorable reconsideration of this application.

The Office rejected claims 1-9 and 11-15 under 35 U.S.C. § 103(a) as obvious over applicant's admitted prior art (AAPA) in view of Milleker. The Office further rejected claim 10 under 35 U.S.C. § 103(a) as obvious over AAPA in view of Milleker and further in view of Francis.

Applicant respectfully traverses the prior art rejections. Particularly, the Office asserted that essentially all of the steps recited in claim 1 are described in Applicant's background section and particularly on page 7, lines 1-15 and in steps 104-107 of Figure 1 except, in the step of building the resource, that the resource is built for containing the requested object, <u>but not containing the requested object</u>.

However, the Office said that this was found in Milleker at column 5, lines 40-50.

The present invention is an efficient method for exchanging data between two computer application programs or between a resource library and an application program. In accordance with a specific exemplary embodiment of the invention, XMI documents for transporting the data between the two applications are built on-the-fly rather than being stored in memory. Further, when a resource library or application program receives a request for an object, the resource library creates the resource to which that object corresponds, but does not populate that resource. Next, the resource library populates the resource with only the object(s) requested. Then the resource is returned to the requesting application program.

This technique is to be contrasted with the prior art technique described in the background section of the present application at page 7, in which the resource repository calls a resource factory that uses a URI converter that takes the URI specified in the request and returns the contents of that URI. Next, the URI converter attempts to open the input stream from the document and return the resulting input stream. Assuming the stream has been successfully retrieved and converted, the receiving entity parses the input stream into the individual objects and writes the resource containing the objects to the cache. Finally, the particular one of the objects in the parsed document that was requested is located in the resource and returned to the tool.

It is important to note that, in this routine, the entire XMI document within which the requested object is contained is retrieved converted, parsed and written to the cache even though only a single object was requested.

The present invention is very different from the prior art described on page 7 of the present application and shown in the flow chart of Figure 1. In the AAPA, the resource factory does not create the resources on-the-fly, nor does it populate the resource with <u>only</u> the object requested. Rather, it stores fully populated resources in memory. It retrieves them when an object in that resource is requested and sends the entire resource to the requesting application program, even if only a single object in that resource was requested. This is wasteful and inefficient.

The present invention solves the aforementioned inefficiency of the AAPA and other problems.

In view of the above discussed differences between the present invention as claimed and the AAPA, it appears that it would be necessary to take rather unusual interpretations of the claim language to conclude that the claim language covers the AAPA. Nevertheless, Applicant has herein amended the independent claims so as to better prevent any possible interpretation of the claims that would read on the AAPA as described by the Office in the latest Office Action.

With reference to exemplary independent claim 1, the AAPA does not have a resource factory "including a plurality of software modules for building resources from a data source". The resources are already built in the AAPA and are merely retrieved responsive to requests for objects within the resources. Applicant assumes that it is the Office's position that this recitation reads on the AAPA because, inherently in the AAPA, the resources had to be built somehow, somewhere at some time prior to their being retrieved. However, this is an improper and strained reading of the claim language. Nevertheless, Applicant has amended the claim to recite "each software module adapted for building resources from a data source <u>responsive to a request for an object of a type to which said resource corresponds</u>, each said software module designed to build a resource of a particular type". The additional language that the building step is performed is responsive to a request for an object, eliminates any possibility of reading on pre-built resources.

Furthermore, the AAPA does not incorporate step (2) of "responsive to a request for an object from a first computing entity, selecting a software module for building a resource of the type to which said requested object corresponds" or step (3) of "building a resource for containing said requested object..., said resource populated with information defining said resource, <u>but not containing said requested object</u>". Applicant has not changed the language of step (2) because this language already expressly included the limitation that the selection of a software module to build the resource is <u>responsive to a request for an object corresponding to that resource</u>. However, Applicant has amended the language of step (3) to expressly recite that it occurs after step (2). This should prevent any overly broad interpretation of the claim in this regard. Specifically, the Office referred to page 7, lines 10-15 of the AAPA as meeting step (3) of claim 1. However, page 7, lines 10-15 describe the parsing of the retrieved resource <u>at the requesting agent</u>, whereas step (3) recites the building of the resource at the resource factory. Page 7, lines 10-15 essentially have nothing to do with what is recited in step (3).

The AAPA also does not teach step (4) of “subsequent to step (3), inserting said requested object into said resource". The fact that step (4) now expressly recites that this step is performed subsequent to steps (2) and (3) prevents any overly broad interpretation that the Office may have previously given this language so as to cover any building of a resource with objects therein at any time.

Accordingly, claim 1 expressly recites many limitations that are not found in the AAPA.

Milliker has been cited for the sole alleged teaching of building the resource first without populating it with objects and then populating the resource with the objects. Thus, Milleker does not add any of the above-discussed teachings lacking from AAPA. Accordingly, independent claim 1 distinguishes over AAPA.

Furthermore, Milleker does not teach that for which it has been cited. Specifically, the Office asserted that Milleker teaches it in column 5, lines 40-50 building a resource first and then inserting an object into it. However, Milleker contains no such disclosure in column 5, lines 40-50. Column 5, lines 40-50 describes the creation of an <u>object</u> and populating the object with <u>attributes</u>, whereas the claim recites creating a <u>resource</u> and inserting into the resource an <u>object</u>. Thus, Milleker does not even teach that for which it has been cited. Milleker describes here nothing but standard OOP protocol.

All of claims 2-9 and 11-15 depend from claim 1 and, therefore, distinguish over AAPA for at least the same reasons.

In view of the foregoing amendments and remarks, this application is now in condition for allowance. Applicant respectfully requests the Examiner to issue a Notice or Allowance at the earliest possible date. The Examiner is invited to contact the Applicant's undersigned counsel by telephone call in order to further the prosecution of this case in any way.

Respectfully submitted,

<u>July 26, 2006</u>

<u>/Theodore Naccarella/</u>
Theodore Naccarella
Registration No. 33,023
Synnestvedt & Lechner LLP
2600 Aramark Tower
1101 Market Street
Philadelphia, PA 19107
Telephone: (215) 923-4466
Facsimile: (215) 923-2189
Attorney for Applicant

TXN:pmf

S:\I\IBM\IBM RALEIGH RSW\PATENTS\P24929 USA\PTO\RESPONSE TO 4-26-06 NONFINAL OFFICE ACTION DOC